07
트랜잭션, 보안, 관리

MySQL로 배우는 데이터베이스 SQL 프로그래밍

목차

학습목표

- ❖ 트랜잭션의 개념을 이해하고 데이터베이스에서 왜 필요한지 알아본다.
- ❖ 트랜잭션 실행 시 동시성 제어가 필요한 이유를 알아보고,
  락킹을 이용한 동시성 제어 기법에 대해 알아본다.
- ❖ 락킹보다 완화된 방법으로 트랜잭션의 동시성을 높이는 트랜잭션
  고립 수준에 대해 알아본다.
- ❖ 데이터베이스 시스템에 문제가 생길 때의 복구 방법을 알아본다.

01. 트랜잭션

1. 트랜잭션의 개념
2. 트랜잭션의 성질
3. 트랜잭션과 DBMS

1. 트랜잭션의 개념

- **트랜잭션(transaction): DBMS에서 데이터를 다루는 논리적인 작업의 단위**

- **데이터베이스에서 트랜잭션을 정의하는 이유**
  - 데이터베이스에서 데이터를 다룰 때 장애가 일어날 때 데이터를 복구하는 작업의 단위가 됨
  - 데이터베이스에서 여러 작업이 동시에 같은 데이터를 다룰 때가 이 작업을 서로 분리하는 단위가 됨

- **트랜잭션은 전체가 수행되거나 또는 전혀 수행되지 않아야 함(all or nothing).**

  예)은행 업무를 보는데 A 계좌(박지성)에서 B 계좌(김연아)로 10,000원을 이체할 경우

```
BEGIN
        ① A 계좌(박지성)에서 10,000원을 인출하는 UPDATE 문
        ② B 계좌(김연아)에    10,000원을 입금하는 UPDATE 문
END
```

1. 트랜잭션의 개념

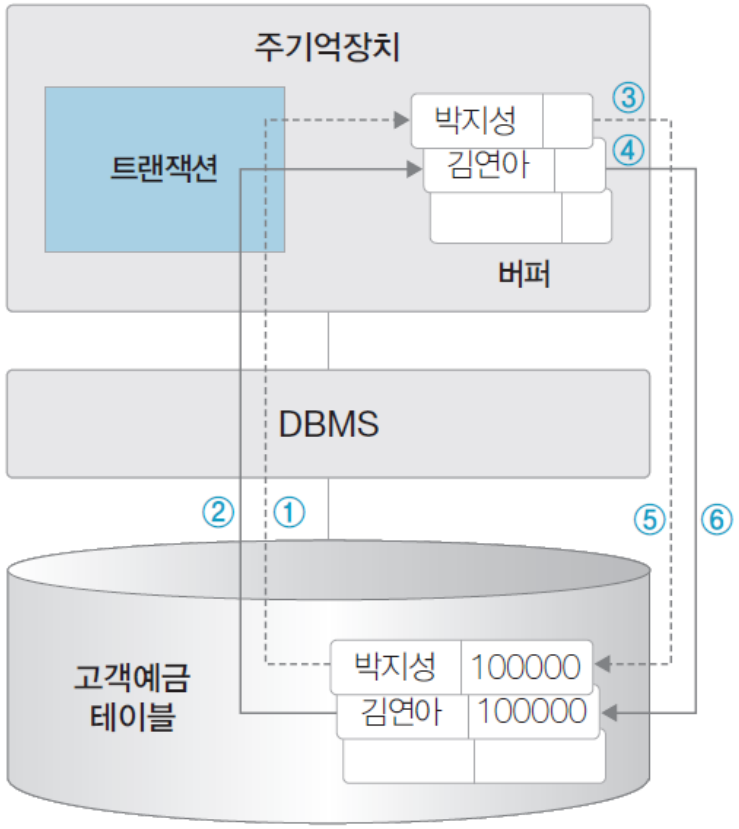
```
START TRANSACTION
① /* 박지성 계좌를 읽어온다 */
② /* 김연아 계좌를 읽어온다 */
   /* 잔고 확인 */

③ /* 예금인출 박지성 */
UPDATE Customer
SET    balance=balance-10000
WHERE  name='박지성';

④ /* 예금입금 김연아 */
UPDATE Customer
SET    balance=balance+10000
WHERE  name='김연아';

COMMIT /* 부분완료 */
⑤ /* 박지성 계좌를 기록한다 */
⑥ /* 김연아 계좌를 기록한다 */
```

(a) 계좌이체 트랜잭션



(b) 트랜잭션 수행 과정

그림 8-1 **계좌이체 트랜잭션과 수행 과정**

1. 트랜잭션의 개념

트랜잭션 수행 과정

① A 계좌(박지성)의 값을 하드디스크(데이터베이스)에서 주기억장치 버퍼로 읽어온다

② B 계좌(김연아)의 값을 하드디스크(데이터베이스)에서 주기억장치 버퍼로 읽어온다.

③ A 계좌(박지성)에서 10,000원을 인출한 값을 저장한다.

④ B 계좌(김연아)에 10,000원을 입금한 값을 저장한다.

⑤ A 계좌(박지성)의 값을 주기억장치 버퍼에서 하드디스크(데이터베이스)에 기록한다.

⑥ B 계좌(김연아)의 값을 주기억장치 버퍼에서 하드디스크(데이터베이스)에 기록한다.

트랜잭션의 종료(COMMIT)를 알리는 방법

- [방법 1] ①-②-③-④-COMMIT-⑤-⑥
- [방법 2] ①-②-③-④-⑤-⑥-COMMIT

→ DBMS는 사용자에게 빠른 응답성을 보장하기 위해 [방법 1]을 선택한다.

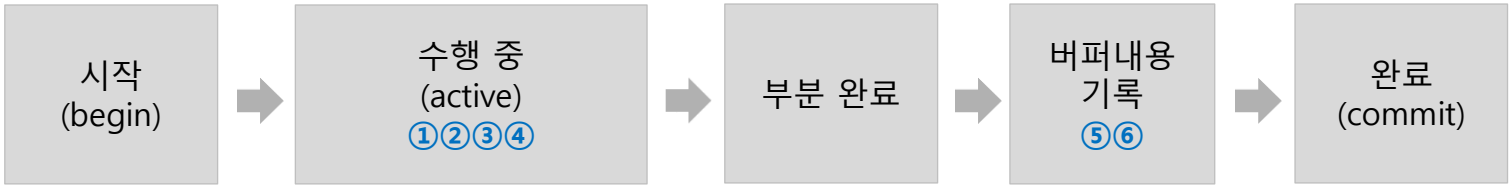


그림 8-2 트랜잭션의 수행 과정

2. 트랜잭션의 성질

표 8-1 트랜잭션과 프로그램의 차이점

구분	트랜잭션	프로그램
프로그램 구조	BEGIN TRANSACTION ... COMMIT TRANSACTION	main() { ... }
다루는 데이터	데이터베이스 저장된 데이터	파일에 저장된 데이터
번역기	DBMS	컴파일러
성질	원자성, 일관성, 고립성, 지속성	-

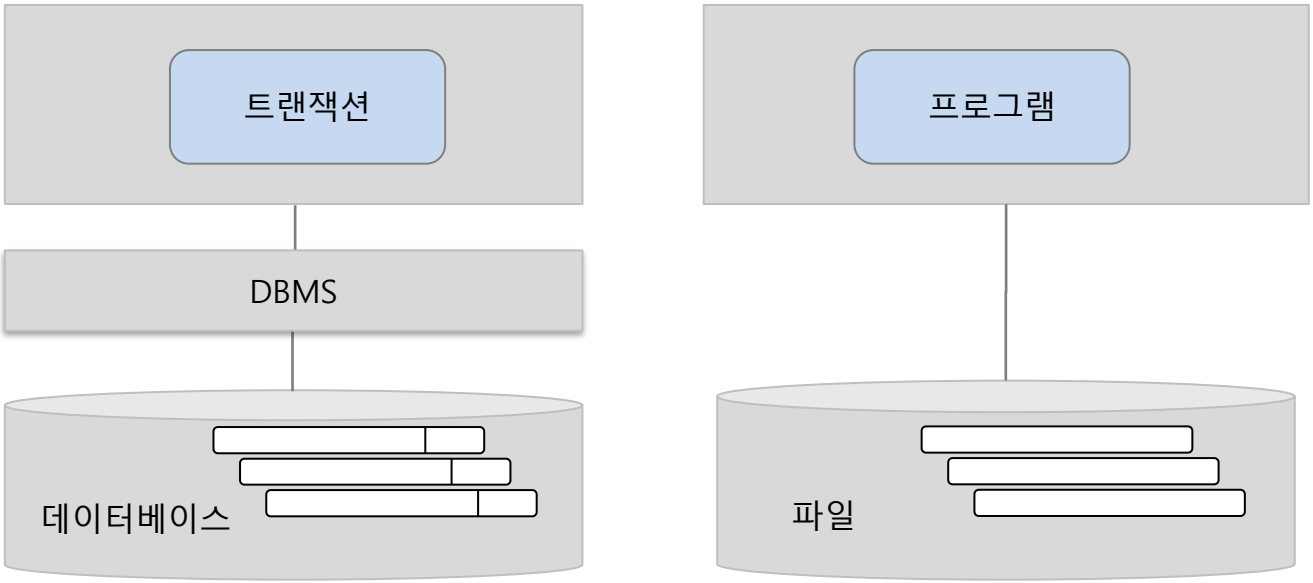


그림 8-3 컴퓨터 시스템 내의 트랜잭션과 프로그램

2. 트랜잭션의 성질

❖ 트랜잭션의 ACID 성질

- **원자성(Atomicity)** : 트랜잭션에 포함된 작업은 전부 수행되거나 아니면 전부 수행되지 않아야 (all or nothing) 함
- **일관성(Consistency)** : 트랜잭션을 수행하기 전이나 수행한 후나 데이터베이스는 항상 일관된 상태를 유지해야 함
- **고립성(Isolation)** : 수행 중인 트랜잭션에 다른 트랜잭션이 끼어들어 변경 중인 데이터 값을 훼손하는 일이 없어야 함
- **지속성(Durability)** : 수행을 성공적으로 완료한 트랜잭션은 변경한 데이터를 영구히 저장해야 함

2. 트랜잭션의 성질

❖ 원자성(Atomicity)

- 트랜잭션이 원자처럼 더 이상 쪼개지지 않는 하나의 프로그램 단위로 동작해야 한다는 의미
- 일부만 수행되는 일이 없도록 전부 수행하거나 아예 수행하지 않아야(all or nothing) 함

표 8-2 **MySQL 트랜잭션 제어 명령어(TCL)**

표준 명령어	MySQL 데이터베이스 문법	설명
START TRANSACTION	SET TRANSACTION NAME <이름>	트랜잭션의 시작
COMMIT	COMMIT	트랜잭션의 종료
ROLLBACK	ROLLBACK {TO <savepoint>}	트랜잭션을 전체 혹은 <savepoint>까지 무효화시킴
SAVEPOINT	SAVEPOINT <savepoint>	<savepoint>를 만듬

2. 트랜잭션의 성질

❖ 일관성(Consistency)

- 트랜잭션은 데이터베이스의 일관성을 유지해야 함.
- 일관성은 테이블이 생성 시 CREATE 문과 ALTER 문의 무결성 제약조건을 통해 명시됨.

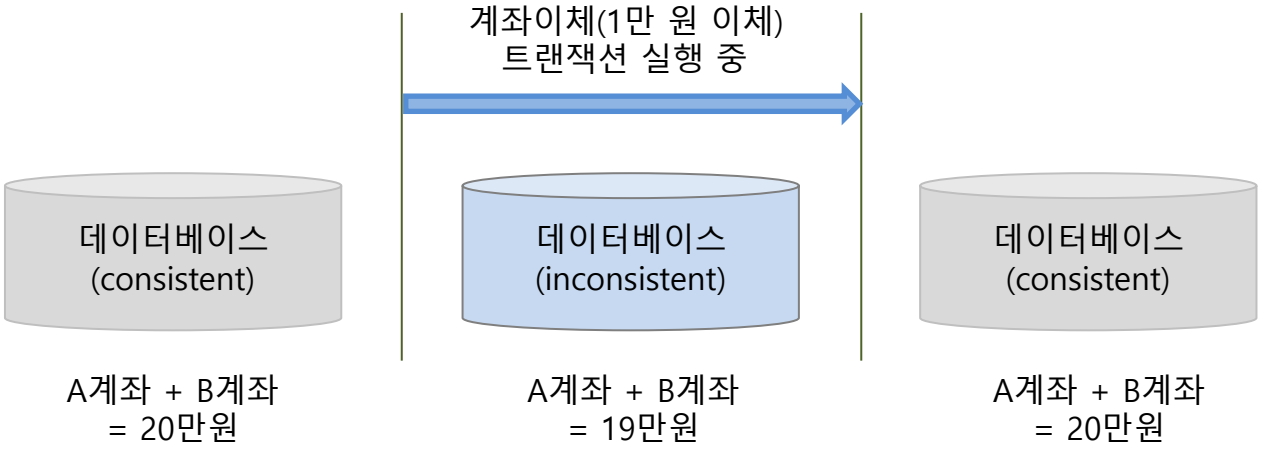


그림 8-4 데이터베이스 변경 중과 변경 후의 일관성

2. 트랜잭션의 성질

❖ 고립성(Isolation)

- 데이터베이스는 공유가 목적이므로 여러 트랜잭션이 동시에 수행됨
- 동시에 수행되는 트랜잭션은 상호 존재를 모르고 독립적으로 수행되는데, 이를 고립성이라고 함.
- 고립성을 유지하기 위해서는 트랜잭션이 변경 중인 임시 데이터를 다른 트랜잭션이 읽고 쓸 때 제어가 필요함.

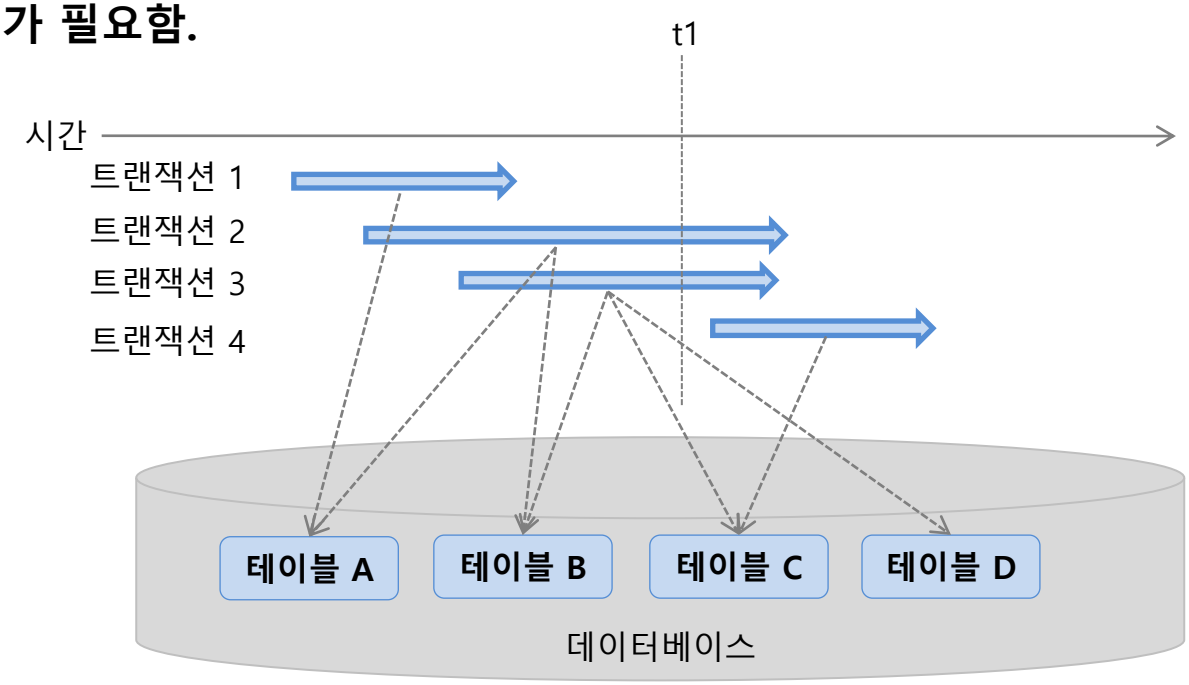


그림 8-5 트랜잭션의 동시 수행과 데이터 공유

2. 트랜잭션의 성질

❖ 지속성(Durability)

- 트랜잭션이 정상적으로 완료(commit) 혹은 부분완료(partial commit)한 데이터는 DBMS가 책임지고 데이터베이스에 기록하는 성질

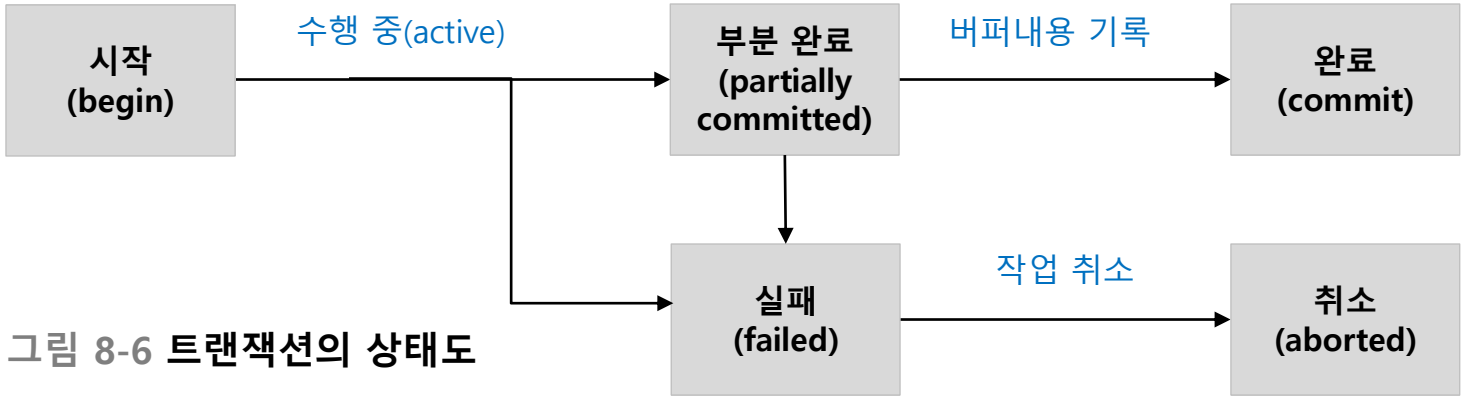


그림 8-6 트랜잭션의 상태도

2. 트랜잭션의 성질

❖ 트랜잭션과 DBMS

- DBMS는 원자성을 유지하기 위해 회복(복구) 관리자 프로그램을 작동시킴
- DBMS는 일관성을 유지하기 위해 무결성 제약조건을 활용함
- DBMS는 고립성을 유지하기 위해 일관성을 유지하는 것과 마찬가지로 동시성 제어 알고리즘을 작동시킴
- DBMS는 지속성을 유지하기 위해 회복 관리자 프로그램을 이용함

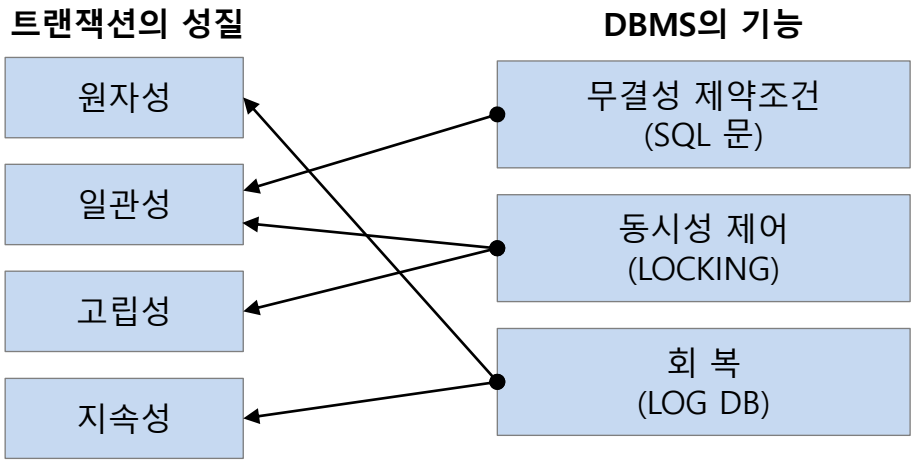


그림 8-7 트랜잭션의 성질과 DBMS의 기능

02. 동시성 제어

1. 갱신손실 문제
2. 락

1. 동시성 제어 개념

- **동시성 제어(concurrency control) :트랜잭션이 동시에 수행될 때, 일관성을 해치지 않도록 트랜잭션의 데이터 접근을 제어하는 DBMS의 기능**

표 8-3 트랜잭션의 읽기(read)/쓰기(write) 시나리오

	트랜잭션1	트랜잭션2	발생 문제	동시접근
[상황 1]	읽기	읽기	읽음(읽기만 하면 아무 문제가 없음)	허용
[상황 2]	읽기	쓰기	오손 읽기, 반복불가능 읽기, 유령 데이터 읽기	허용 혹은 불가 선택
[상황 3]	쓰기	쓰기	갱신손실(절대 허용하면 안 됨)	허용불가(LOCK을 이용)

2. 갱신손실 문제

- **갱신손실(lost update): 두 개의 트랜잭션이 한 개의 데이터를 동시에 갱신(update) 할 때 발생하며, 데이터베이스에서 절대 발생하면 안 되는 현상**

- **[작업 설명] 한 개의 데이터에 두 개의 트랜잭션이 접근하여 갱신하는 작업**
- **[시나리오] 두 개의 트랜잭션이 동시에 작업을 진행**
- **[문제 발생] 갱신손실**
  - T2는 잘못된 데이터로 작업하여 잘못된 결과를 만든 다음, T1의 갱신 작업을 무효화하고 덧쓰기를 수행한 것임
  - T1의 갱신이 손실된 갱신손실(lost update) 문제가 발생한 것임

2. 갱신손실 문제

트랜잭션 T1	트랜잭션 T2	버퍼의 데이터 값
A=read_item(X); ① A=A-100;		X=1000
	B=read_item(X); ② B=B+100;	X=1000
write_item(A → X); ③		X=900
	write_item(B → X); ④	X=1100

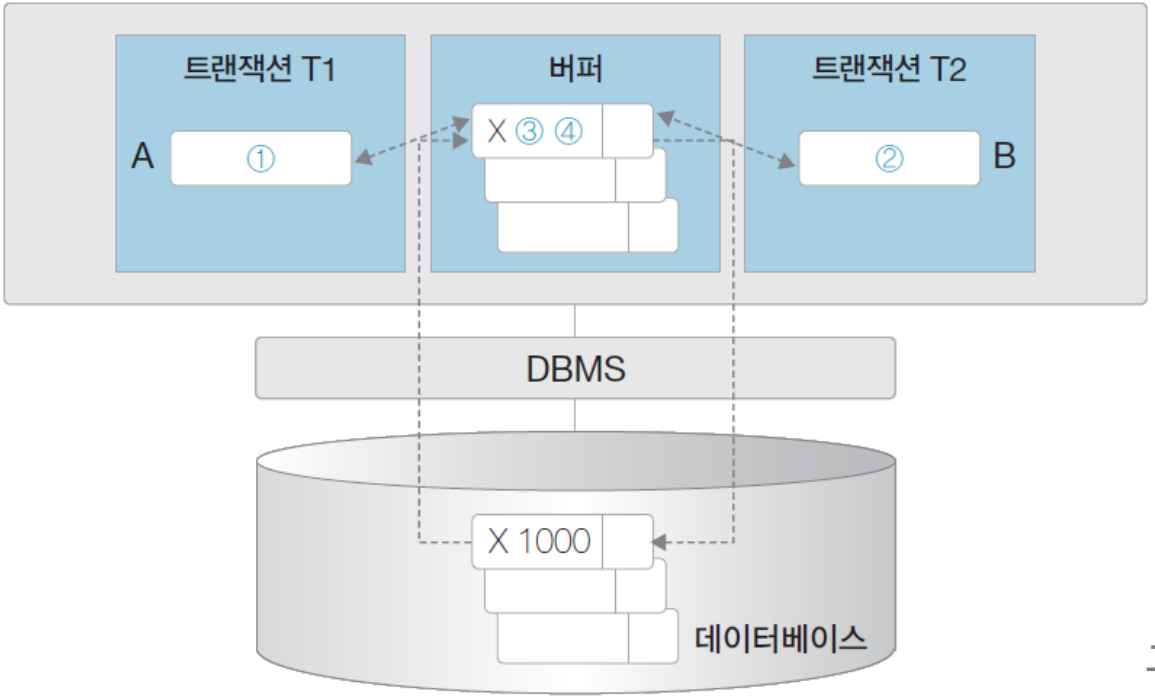


그림 8-8 갱신손실 문제 발생 시나리오

3. 락

- 갱신손실 문제를 해결하려면 상대방 트랜잭션이 데이터를 사용하는지 여부를 알 수 있는 규칙이 필요함.
- 데이터를 수정 중이라는 사실을 알리는 방법의 잠금 장치

3. 락

❖ 락의 개념

트랜잭션 T1	트랜잭션 T2	버퍼의 데이터 값
LOCK(X) A=read_item(X); ① A=A-100;		X=1000
	LOCK(X) (wait... 대기)	X=1000
write_item(A→X); ② UNLOCK(X);		X=900
	B=read_item(X); ③ B=B+100; write_item(B→X); ④ UNLOCK(X)	X=1000

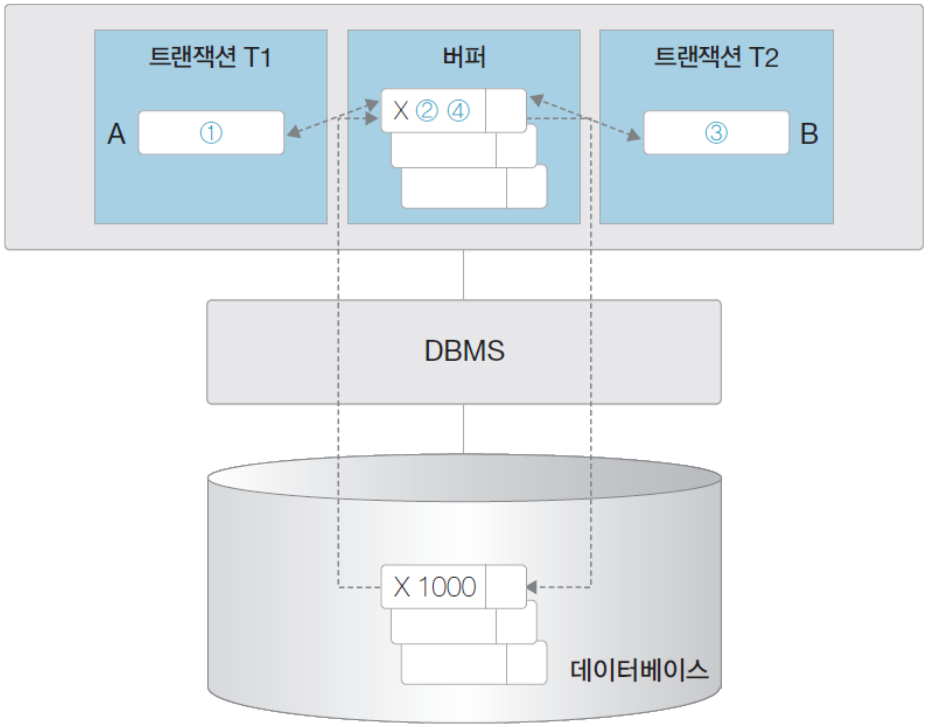


그림 8-9 락을 이용한 갱신손실 문제 해결

3. 락

❖ 락의 개념

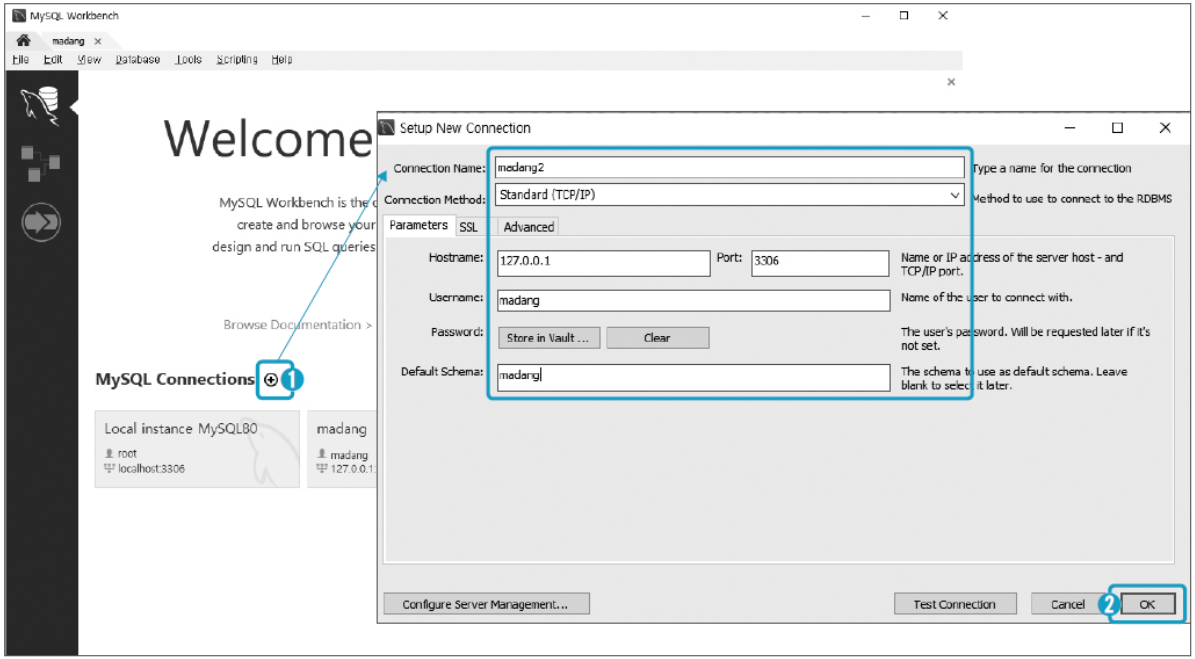
- **[작업 설명] 한 개의 데이터에 두 개의 트랜잭션이 접근하여 갱신하는 작업**

트랜잭션 T1	트랜잭션 T2
START TRANSACTION; USE madang; SELECT * FROM Book WHERE bookid=1; UPDATE Book SET price=7100 WHERE bookid=1; SELECT * FROM Book WHERE bookid=1; COMMIT;	START TRANSACTION; USE madang; SELECT * FROM Book WHERE bookid=1; UPDATE Book SET price=price+100 WHERE bookid=1; SELECT * FROM Book WHERE bookid=1; COMMIT;

여기서 잠깐 실습 시 주의사항

❖ MySQL에서 두 트랜잭션을 동시에 실행시키는 방법

- 트랜잭션 실습을 진행하기 위해서는 MySQL 접속 시 서로 다른 두 세션(세션은 데이터베이스 접속단위)에서 진행해야 한다.
- MySQL을 두 번 연결하기 위해서는 MySQL Workbench에서 접속을 2개 따로 만들어야 한다(madang과 동일한 내용으로 madang2를 만듦).
- 워크시트를 md_madang과 md_madang2로 두 번 접속하여 진행할 수 있다.

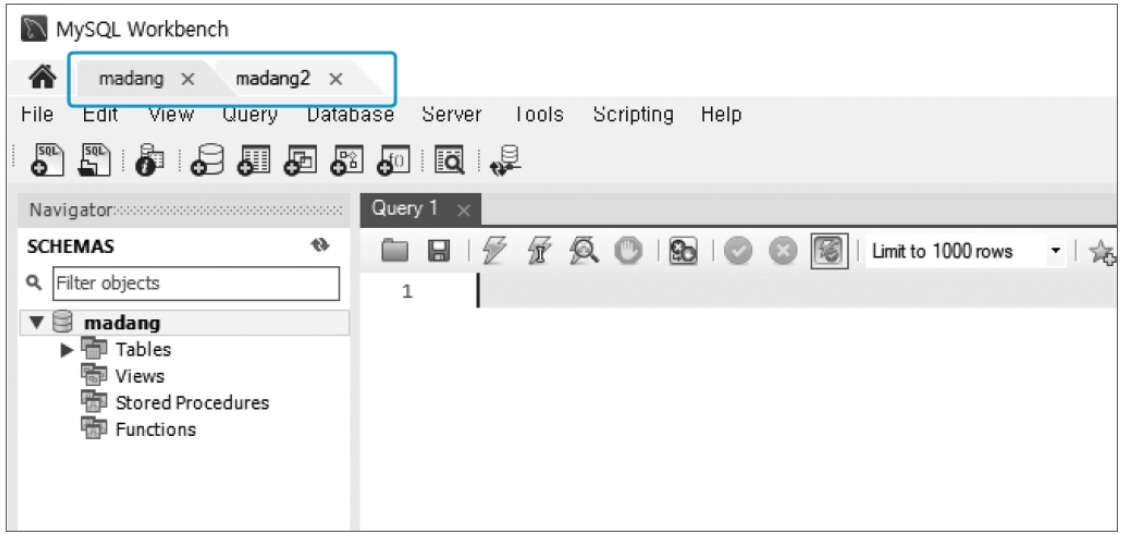


여기서 잠깐 실습 시 주의사항

❖ MySQL에서 두 트랜잭션을 동시에 실행시키는 방법

❶ 두 번째 세션을 위해 Workbench 초기화면에서 madang2 접속을 만든다.

❷ 실습을 위해 madang, madang2 접속을 만든다.



3. 락

❖ 락의 개념

■ [시나리오] 두 트랜잭션을 동시에 실행

트랜잭션 T1	트랜잭션 T2
START TRANSACTION; USE madang; SELECT * FROM Book WHERE bookid=1; UPDATE Book SET price=7100 WHERE bookid=1;	
	START TRANSACTION; USE madang; SELECT * FROM Book WHERE bookid=1; UPDATE Book SET price=price+100 WHERE bookid=1;

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7000

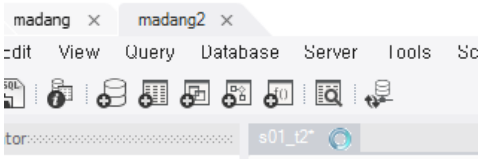
3. 락

❖ 락의 개념

■ [시나리오] 두 트랜잭션을 동시에 실행

트랜잭션 T1	트랜잭션 T2
SELECT * FROM Book WHERE bookid=1; COMMIT;	...(30초간 대기)...
	SELECT * FROM Book WHERE bookid=1; COMMIT;

bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7100



bookid	bookname	publisher	price
1	축구의 역사	굿스포츠	7200

3. 락

❖ 락의 유형

- **공유락(LS, shared lock): 트랜잭션이 읽기를 할 때 사용하는 락**
- **배타락(LX, exclusive lock): 읽고 쓰기를 할 때 사용하는 락**

- **공유락과 배타락을 사용하는 규칙**
  - 데이터에 락이 걸려있지 않으면 트랜잭션은 데이터에 락을 걸 수 있다.
  - 트랜잭션이 데이터 X를 읽기만 할 경우 LS(X)를 요청하고, 읽거나 쓰기를 할 경우 LX(X)를 요청한다.
  - 다른 트랜잭션이 데이터에 LS(X)을 걸어둔 경우, LS(X)의 요청은 허용하고 LX(X)는 허용하지 않는다.
  - 다른 트랜잭션이 데이터에 LX(X)을 걸어둔 경우, LS(X)와 LX(X) 모두 허용하지 않는다.
  - 트랜잭션이 락을 허용받지 못하면 대기 상태가 된다.

표 8-4 락 호환행렬

요청 \ 상태	LS 상태	LX 상태
LS 요청	허용	대기
LX 요청	대기	대기

3. 락

❖ 2단계 락킹

- **2단계 락킹 기법 : 락을 걸고 해제하는 시점에 제한을 두지 않으면 두 개의 트랜잭션이 동시에 실행될 때 데이터의 일관성이 깨질 수 있어 이를 방지하는 방법**
  - 확장단계(Growing phase, Expanding phase) : 트랜잭션이 필요한 락을 획득하는 단계로, 이 단계에서는 이미 획득한 락을 해제하지 않음
  - 수축단계(Shrinking phase) : 트랜잭션이 락을 해제하는 단계로, 이 단계에서는 새로운 락을 획득하지 않음.

- **[작업 설명] 두 개의 데이터에 두 개의 트랜잭션이 접근하여 갱신하는 작업**
- **[문제 발생] 락을 사용하되 2단계 락킹 기법을 사용하지 않을 경우**
- **[문제 해결] 2단계 락킹 기법을 사용할 경우**

3. 락

❖ 2단계 락킹

- **[문제 발생] 락을 사용하되 2단계 락킹 기법을 사용하지 않을 경우**

트랜잭션 T1	트랜잭션 T2	A, B 값

LX(A)
t1=read_item(A);
t1=t1-100;
A=write_item(t1);
UN(A)
		A=900
		B=1000

	LX(A)
	t2=read_item(A);
	t2=t2*1.1;
	A=write_item(t2);
	UN(A)
	LX(B)
	t2=read_item(B);
	t2=t2*1.1;
	B=write_item(t2);
	UN(B)
		A=990
		B=1100

LX(B)
t1=read_item(B);
t1=t1+100;
B=write_item(t1);
UN(B)
		A=990
		B=1200
		/* A+B=2190이므로 일관성 제약 조건에 위배됨 */

3. 락

❖ 2단계 락킹

■ [문제 해결] 2단계 락킹 기법을 사용할 경우

트랜잭션 T1	트랜잭션 T2	A, B 값

LX(A)
t1=read_item(A);
t1=t1-100;
A=write_item(t1);
		A=900
		B=1000

	LX(A) ...(대기상태)...	

LX(B)
t1=read_item(B);
t1=t1+100;
B=write_item(t1);
UN(A)
UN(B)
		A=900
		B=1100

	LX(A)
	t2=read_item(A);
	t2=t2*1.1;
	A=write_item(t2);
	LX(B)
	t2=read_item(B);
	t2=t2*1.1;
	B=write_item(t2);
	UN(A)
	UN(B)
		A=990
		B=1210
		/* A+B=2200이므로 일관성 제약 조건을 지킴 */

3. 락

❖ 데드락

- **두 개 이상의 트랜잭션이 각각 자신의 데이터에 대하여 락을 획득하고 상대방 데이터에 대해 락을 요청하면 무한 대기 상태에 빠질 수 있는 현상으로, 교착상태라고도 함**

- **[작업 설명] 두 개의 데이터에 두 개의 트랜잭션이 접근하여 갱신하는 작업**
- **[문제 발생] Oracle DBMS에서 데드락 발생**
- **[문제 해결] 데드락 해결**
  - 일반적으로 데드락이 발생하면 DBMS는 T1 혹은 T2의 작업 중 하나를 강제로 중지시킴
  - 그 결과 나머지 트랜잭션은 정상적으로 실행됨
  - 이때 중지시키는 트랜잭션에서 변경한 데이터는 원래 상태로 되돌려 놓음

2. 락

트랜잭션 T1	트랜잭션 T2
START TRANSACTION; USE madang; UPDATE Book SET price=price+100 WHERE bookid=1;	
	START TRANSACTION; USE madang; UPDATE Book SET price=price*1.1 WHERE bookid=2;
UPDATE Book SET price=price+100 WHERE bookid=2; ...(대기상태)...	
	UPDATE Book SET price=price*1.1 WHERE bookid=1; -- Error Code: 1213. Deadlock found when trying to get lock; try restarting transaction
COMMIT;	COMMIT;

3. 락

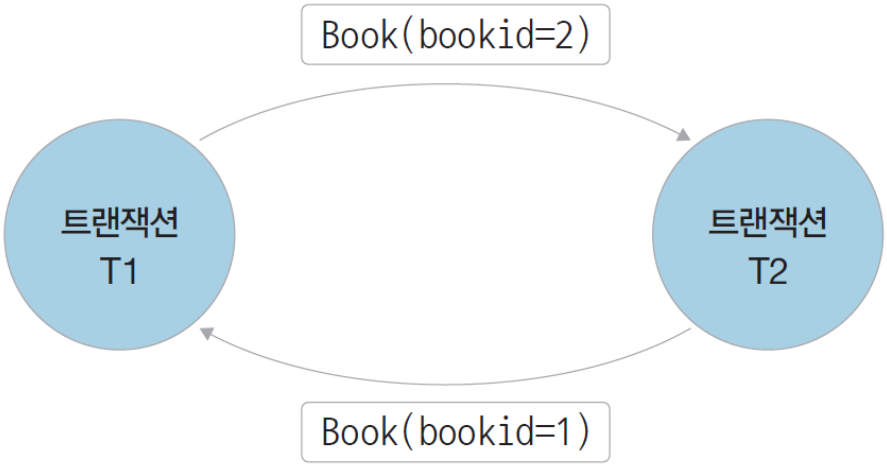


그림 8-10 대기 그래프

03. 보안과 권한

1. 로그인 사용자 관리
2. 권한 관리
3. 권한 관리 실습

보안과 권한

- DBMS는 ① 로그인 단계에서 DBMS 접근을 제한하는 로그인 사용자 관리
  ② 로그인한 사용자별로 특정 데이터로의 접근을 제한하는 권한 관리의 기능 제공

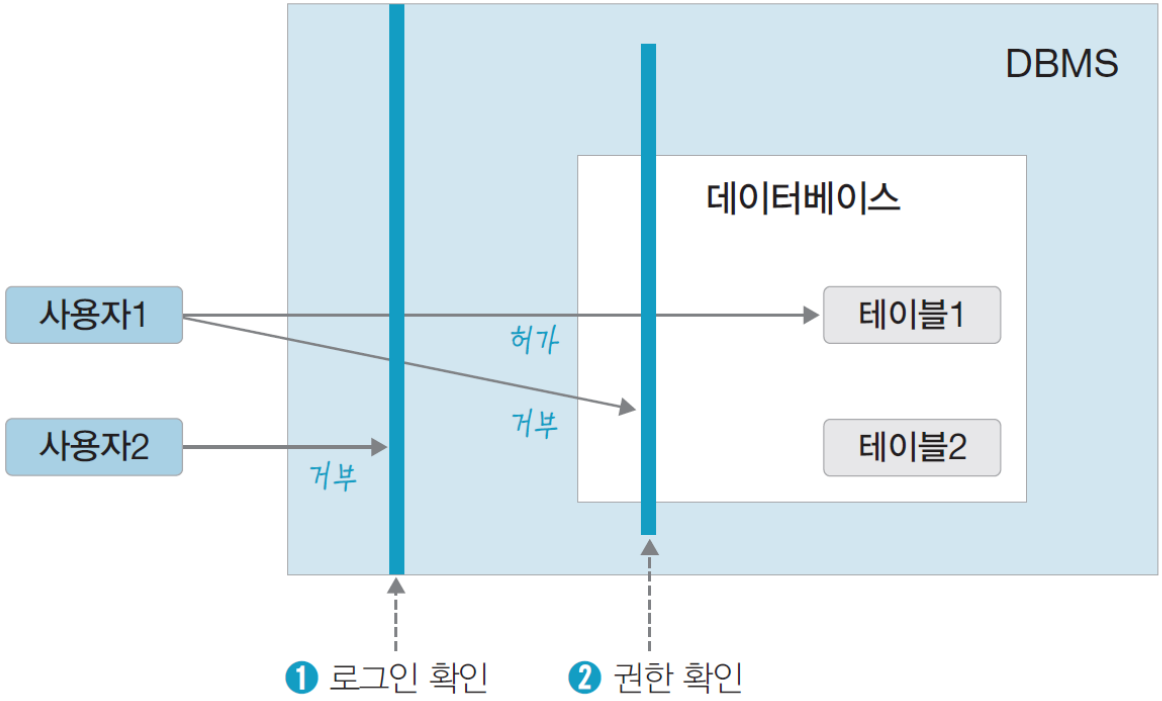


그림 9-2 데이터베이스 접근 권한

1. 로그인 사용자 관리

❖ 사용자 계정 생성

- **CREATE USER [사용자이름] IDENTIFIED BY [비밀번호];**

(예) localhost에서 접속하는 madang 사용자 생성

localhost에서 접속하는 madang 사용자는 다음과 같이 생성한다.

비밀번호는 IDENTIFIED BY 다음에 있는 'madang'이다.

```
CREATE USER madang@localhost IDENTIFIED BY 'madang';
```

외부에서 접속 가능한 madang 사용자는 다음과 같이 생성한다.

```
CREATE USER madang@'%' IDENTIFIED BY 'madang';
```

특정 사이트 'happy.md.kr' 사이트에서 접속 가능한 madang 사용자 계정은 다음과 같이 생성한다.

```
CREATE USER madang@'%.happy.md.kr' IDENTIFIED by 'madang';
```

1. 로그인 사용자 관리

❖ 사용자 계정 삭제

- **DROP USER [사용자이름] CASCADE;**

2. 권한 관리

- 소유한 개체에 대한 사용 권한을 관리하기 위한 명령을 DCL(Data Control Language)이라고 함
- 대표적 DCL 문 : 권한 허가 GRANT 문, 권한 취소 REVOKE 문

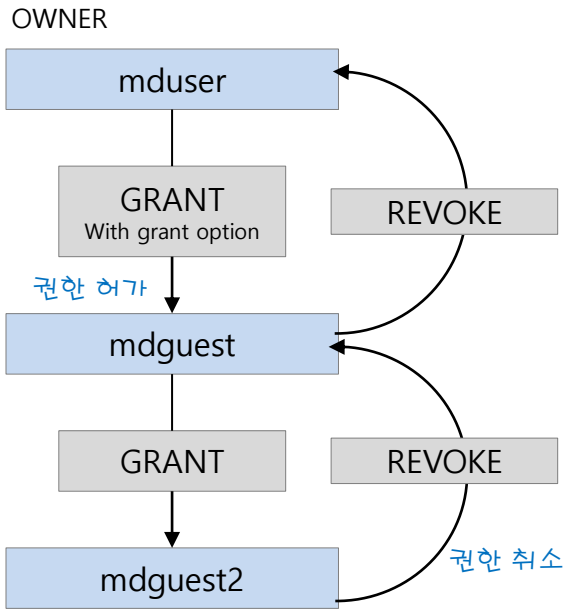


그림 9-4 GRANT 문과 REVOKE 문의 관계

3. 권한 관리 실습

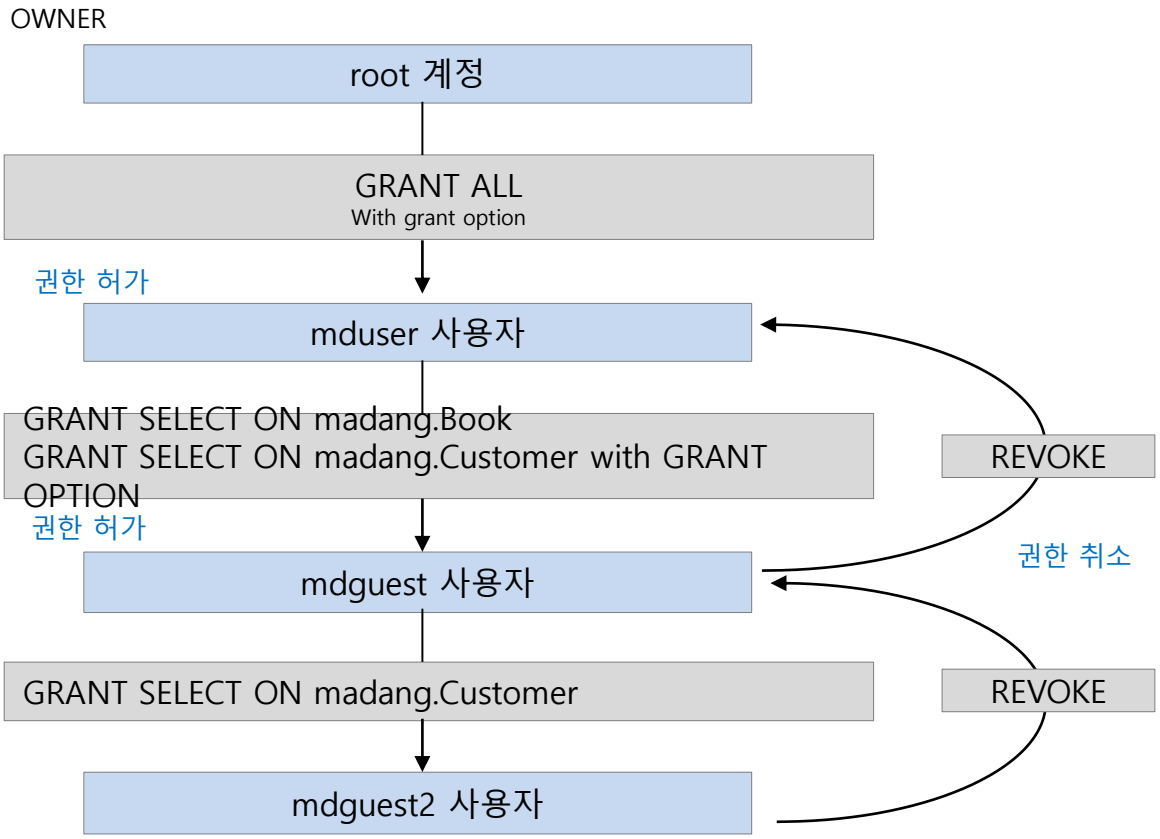


그림 9-5 **GRANT 문과 REVOKE 문 실습**

3. 권한 관리 실습

❖ 권한 허가 - GRANT

객체를 생성한 소유자가 대상 객체에 대한 권한을 다른 사용자에게 허가하는 명령

```
GRANT 권한 [(컬럼[ ,...n ])] [ ,...n ]
        [ON 객체] TO {사용자I롤IPUBLIC [ ,...n ]}
        [WITH GRANT OPTION]
```

* [, ... n] : 반복가능을 의미

질의 9-5 (madang 계정) mdguest에게 Book 테이블의 SELECT 권한을 부여하시오.

```
GRANT SELECT ON madang.Book TO mdguest@localhost;
```

0 row(s) affected

질의 9-6 (madang 계정) mdguestguest에게 Customer 테이블의 SELECT, UPDATE 권한을 WITH GRANT OPTION과 함께 부여하시오.

```
GRANT SELECT, UPDATE ON madang.Customer TO mdguest@localhost
        WITH GRANT OPTION;
```

GRANT SELECT, UPDATE ON madang.Customer TO mdguest@loca... 0 row(s) affected

3. 권한 관리 실습

❖ 권한 허가 - GRANT

질의 9-7 (mdguest 계정) madang.Book 테이블과 madang.Customer 테이블의 SELECT 권한을 mdguest2에 부여하시오.

GRANT SELECT ON madang.Book TO mdguest2;

```
GRANT SELECT ON madang.Book ...  Error Code: 1142. GRANT command denied to user 'mdguest'@'localhost' for table 'book'
```

GRANT SELECT ON madang.Customer TO mdguest2;

```
GRANT SELECT ON madang.Custo...  0 row(s) affected
```

3. 권한 관리 실습

❖ 권한 취소 - REVOKE

- **GRANT 문으로 허가한 권한을 취소, 회수하는 명령**

```
REVOKE 권한 [(컬럼[ ,...n ])] [ ,...n ]
    [ON 객체] FROM { 사용자I롤IPUBLIC [ ,...n ]}
    [CASCADE]
```

- **GRANT 문이 권한 부여를 위해 'TO 사용자'를 표기했다면, REVOKE 문은 권한 취소를 위해 'FROM 사용자'를 표기함**
- **권한을 재부여하는 WITH GRANT OPTION의 회수를 위해 'CASCADE' 옵션을 사용함**
- **CASCADE는 사용자가 다른 사용자에게 부여한 권한까지 연쇄적으로 취소하라는 의미로, 사전에 주의 깊게 확인하고 사용해야 함**

3. 권한 관리 실습

❖ 권한 취소 - REVOKE

질의 9-8 (madang 계정) mdguest에게 부여된 Book 테이블의 SELECT 권한을 취소하시오.

REVOKE SELECT ON madang.Book FROM mdguest@localhost;

```
REVOKE SELECT ON madang.Book FROM mdguest@localhost          0 row(s) affected
```

질의 9-9 (madang 계정) mdguest에게 부여된 Customer 테이블의 SELECT 권한을 취소하시오.

REVOKE SELECT ON Customer FROM mdguest;

```
REVOKE SELECT ON madang.Customer FROM mdguest@localhost     0 row(s) affected
```

04. 백업과 복원

1. 백업의 종류
2. 데이터베이스 백업 방법
3. MySQL 백업 및 복원 실습

백업과 복원

- 백업(backup) : 데이터베이스에서도 역시 예상하지 못한 장애에 대비하여 데이터베이스를 복제하여 보관하는 작업
- 복원(recovery) :장애가 발생하여 운영 중인 데이터에 손상이 발생했을 때 기존에 복사해 둔 백업 파일을 사용하여 원래대로 되돌려 놓는 작업
- 미디어 오류
- 사용자 오류
- 하드웨어 장애

1. 백업의 종류

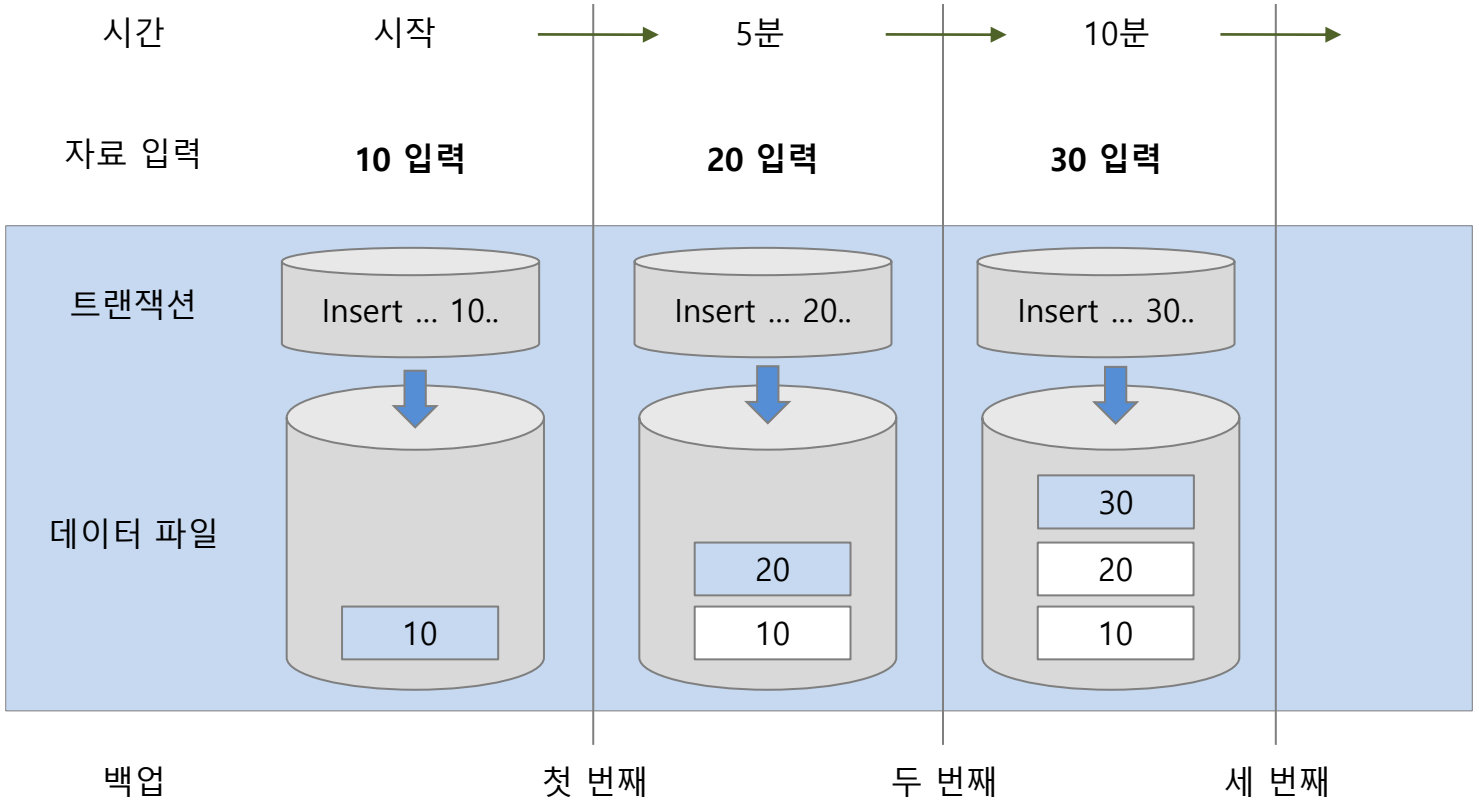


그림 9-6 **데이터 파일의 입력 순서(시간순)**

1. 백업의 종류

- 전체 백업
- 차등 백업
- 트랜잭션 로그 백업

2. 데이터베이스 백업 방법

❖ 물리적 백업

데이터베이스를 구동하기 위해 필요한 모든 파일을 물리적으로 '복사'하는 방법

- 콜드 백업 : 데이터베이스를 셧다운(shutdown)한 후에 백업을 진행하는 방법
- 핫 백업 : 운영 중인 데이터베이스의 파일을 복사하는 방법

❖ 논리적 백업

실제 데이터베이스를 구성하는 물리적 파일을 직접 복사하는 방법이 아닌,
데이터베이스의 콘텐츠(내용)를 별도의 파일로 옮기는 백업 방법

3. MySQL 백업 및 복원 실습

❖ 기본 준비

❶ 실제 백업된 파일이 저장될 폴더 준비 -> C:₩madang₩mdbackup 폴더 생성

❷ Workbench에서 root 계정으로 쿼리 창을 만듦. 여기서는 madang 데이터베이스 전체를 백업하고 복원해보기로 함

❸ 백업이 끝나면 백업 화면을 닫고 백업된 파일을 살펴본다.
C:\madang\backup\madang_backup.sql 경로에서 파일 확인

3. MySQL 백업 및 복원 실습

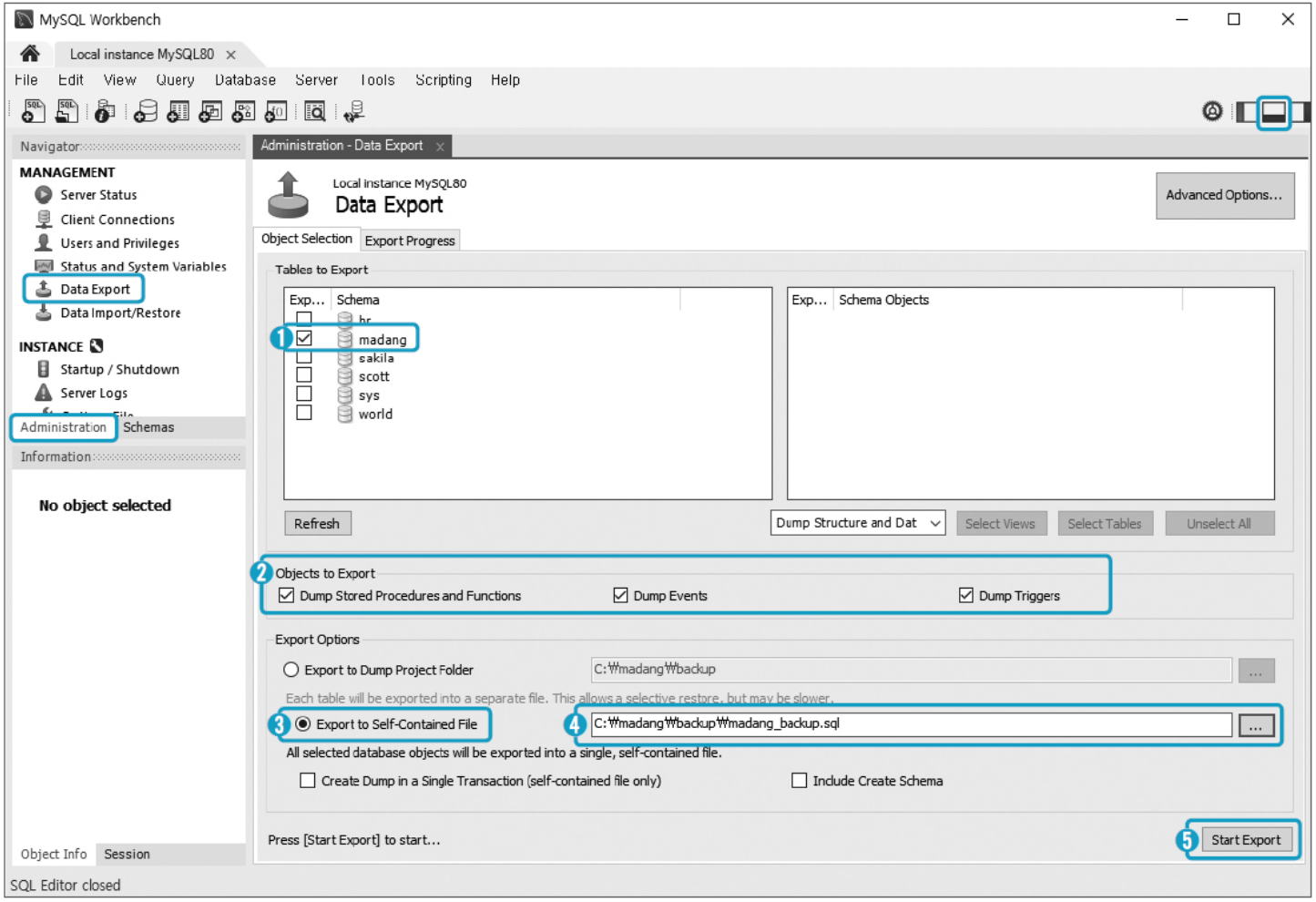


그림 9-7 데이터베이스 백업 – Export 화면

3. MySQL 백업 및 복원 실습

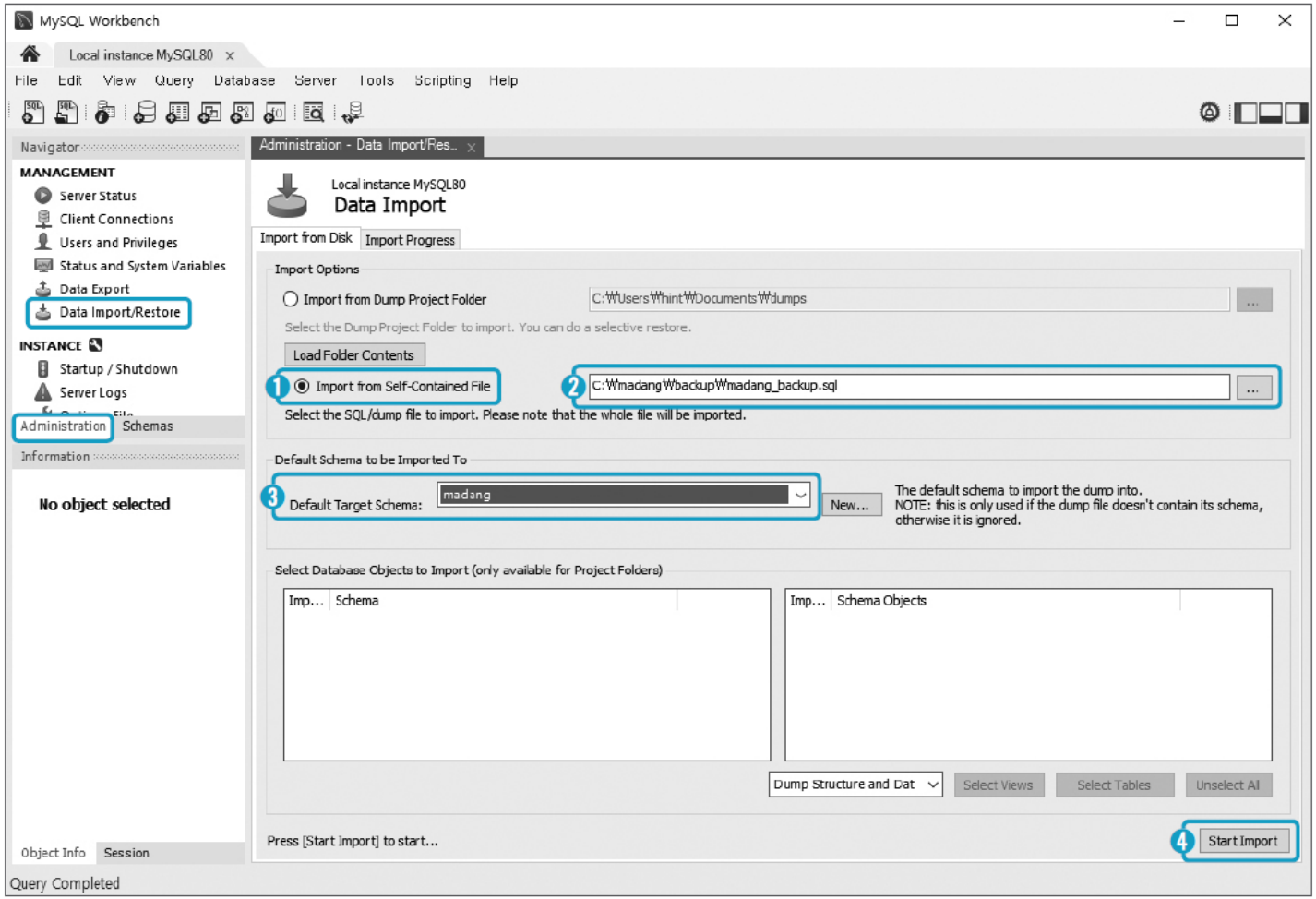


그림 9-8 데이터베이스 복원 – Import 화면